In the Specification:

The paragraph spanning page 1, lincs 2 through 10 has been amended as shown below:

The present application claims benefit of U.S. Provisional Application No. 60/252,092, filed November 21, 2000. This patent application is also related to the following two co-pending and commonly-owned applications:

1. R. Mech, "Rendering Volumetric Fog and Other Gaseous Phenomena", U.S. Patent Application Serial No. 09,990,085 ~~(MS1-1031US)~~ filed concurrently herewith on November 21, 2001 and ~~(~~incorporated in its entirety herein by reference~~)~~; and

A1

2. R. Mech, "Method, System, and Computer Program Product for Rendering Multicolored Layered Fog with Self-Shadowing and Scene Shadowing", U.S. Patent Application Serial No. 09/990,082 ~~(MS1-1033US)~~, filed concurrently herewith on November 21, 2001 and ~~(~~incorporated in its entirety herein by reference~~)~~.

The paragraph spanning page 2, line 13 through page 3, line 2, has been amended as shown below:

One approach renders clouds and light shafts by blending a set of billboards, representing metaballs. The rendering times range ~~ranges~~ from 10 to 30 seconds for relatively small images. Another approach renders gases by blending slices of the volume in the view direction. Using 3D textures, a near real-time frame rate is achieved. This is especially true for smaller images. Unfortunately, both these techniques are fill-limited, i.e., the number and the size of the rendered semi-transparent polygons is limited by the number of pixels hardware can render per second. Even on the fastest machines it is not possible to render too many full-screen polygons per frame. The techniques may be suitable for rendering smaller local objects, e.g., a smoke column or a cloud, but even then the performance can suffer when the viewer comes too close to the object and the semitransparent polygons fill the whole screen. In the case of a patchy fog that can be spread across a large part of the scene the number of slices would be simply too large.

The paragraph on page 3, spanning lines 3 through 13, has been amended as shown below:

In real-time animations, smoke and clouds are usually simulated by mapping transparent textures on a polygonal object that approximates the boundary of the gas. Although the texture may simulate different densities of the gas inside the 3D boundary and compute even the light scattering inside the gas, it does not change, when viewed from different directions, and it does not allow movement through the gas without sharp transitions. Consequently, these techniques are suitable for rendering very dense gases or gases ~~gasses~~ viewed from a distance. Other methods simplify their task by assuming constant density of the gas at a given elevation, thereby making it possible to use 3D textures to render the gas in real time. The assumption, however, prevents using the algorithm to render patchy fog.

A3

The paragraph on page 5, spanning lines 13-18, has been amended as shown below:

Once the travel distance information is obtained in the alpha buffer, the method converts the travel distance information to a fog factor. A scene is then rendered based on a blending of the scene color and the fog factor. In this way, a computer graphics image of a scene is displayed that depicts the volumetric fog or other gaseous phenomena ~~fast~~ rapidly, with high quality realism. This approach can be carried out quickly at a real-time ~~rate~~ rates.

A4

The paragraph on page 5, spanning lines 19-25, has been amended as shown below:

In another described implementation a multipass routine includes pixel texture processing steps that can be carried out in an OPENGL® graphics environment having a pixel texture extension. This OPENGL® graphics environment includes a texture coordinate generator (also called texgen). In this environment, the method includes steps of initializing the texgen and setting a one-dimensional texture to a range of texel values extending from a minimum value to a maximum value.

A5

The paragraph on page 6, spanning lines 11-20, has been amended as shown below:

In one system implementation, a one-dimensional texture is stored with texels having values ranging from a minimum to a maximum (e.g., from 0 to 255 if 8 bits are used). The texture coordinate generator receives a distance value (such as, a pixel coordinate or a coordinate of a boundary point of a volume region) and generates a one or more corresponding one-dimensional texture coordinates ~~coordinate (s)~~. The one or more dimensional texture coordinates ~~coordinate (s) is~~ may be used to sample a respective texel in the one-dimensional texture. The sampled texel value is then stored in an alpha channel of an alpha buffer corresponding to a respective pixel. In this way, distance information can be processed in pixel texture processing hardware with an output provided to an alpha buffer.

A6

The paragraph spanning page 7, line 9, through page 8, line 3 has been amended as shown below:

FIG. 1 illustrates a block diagram of an ~~example~~ exemplary computer architecture 100 implementation. Architecture 100 includes six overlapping layers. Layer 110 represents a high level software application program. Examples of such software application programs include visual simulation applications, computer games or any other application that could be made to take advantage of computer generated graphics. Layer 120 represents a three-dimensional (3D) graphics software tool kit, such as OPENGL® PERFORMER, available from Silicon Graphics, Incorporated, Mountain View, California. Layer 125 represents a graphics application program interface (API), which can include but is not limited to OPENGL® software, available from Silicon Graphics, Incorporated. Layer 130 represents system support such as operating system and/or windowing system support. ~~Two examples~~ Examples of such support systems include the LINUX®, UNIX® and Windows® operating/windowing systems. Layer 135 represents firmware which can include proprietary computer code. Finally, layer 140 represents hardware, including graphics hardware. Hardware can be any hardware or graphics hardware including, but not limited to, a computer graphics processor (single chip or multiple chip), a specially designed computer, an interactive graphics machine, a gaming platform, a low end game system, a game console, a network architecture, server, et cetera. Some or all of the layers 110-140 of architecture 100 will be available in most commercially available computers.

The paragraph on page 11, spanning lines 1-8, has been amended as shown below:

In alternative ~~implemenations~~ implementations, secondary memory 314 may include other similar means for allowing computer programs or other instructions to be loaded into computer system 300. Such means can include, for example, a removable storage unit 324 and an interface 322. Examples can include a program cartridge and cartridge interface (such as that found in video game devices), a removable memory chip (such as an EPROM, or PROM) and associated socket, and other removable storage units 324 and interfaces 322 which allow software and data to be transferred from the removable storage unit 324 to computer system 300.

The paragraph on page 14, spanning lines 1-14, has been amended as shown below:

In step 420, travel distance information is obtained in an alpha channel. "Travel distance information" refers to any function of component distances that are defined with respect to respective three-dimensional bounded volume regions covered by a pixel. "Scaled traveled distance information" refers to travel distance information which is scaled. "Total travel distance information" refers to a sum of distances through each three-dimensional bounded volume region along a ray between a respective pixel and a reference point. "Scaled traveled distance information" refers to total travel distance information which is scaled. Travel distance information obtained in an alpha channel can include, but is not limited to, travel distance information, scaled travel distance information, total travel distance information, and/or scaled total travel distance information Obtaining total travel distance information in an alpha channel in step 420 is described in further detail below with respect to implementations in FIGs. 5A, 5B, 6 and 7A-~~7G~~ 7F.

The paragraph spanning page 14, line 15, through page 15, line 4 has been amended as shown below:

In step 430, travel distance information in the alpha channel is then converted to a fog factor (also called an attenuation factor). In step 440, a blending operation blends scene color with fog color based on the fog factor output from step 430. In this way, scene color is blended with fog color to generate a final display image. Any conventional fog equation and blending operation can be used to implement steps 430 and 440 as would be apparent to a person skilled in the art given this description. See, e.g., similar steps for converting travel distance information to a fog factor and blending a scene color and fog color as described in the co-pending application by R. Mech, "Rendering Volumetric Fog and Other Gaseous Phenomena", filed concurrently herewith and incorporated in its entirety herein by reference~~)~~. The display image includes the scene and a simulation of gaseous phenomena in the volume regions. For example, the scene can include a simulation of patchy fog, clouds, or other gases. In one example, the gases have a constant density and color. In other examples, the gases can vary in density and color.

The paragraph on page 15, spanning lines 5-16, has been amended as shown below:

In one example implementation shown in FIGs. 5A and 5B, step 420 is implemented as a multipass routine (steps 502-544). Step 526 is shown in further detail with respect to an example implementation in FIG. 6. To further illustrate the operation of the multipass routine, reference is also made to an example in FIGs. 7A-7F. This example shows two pixels P1, P2 being rendered in a scene having three volume regions 710-730 (also called "fog areas" or "fog regions"). For instance, FIG. 7A shows an example of the output of previous step 412 where a scene is drawn relative to an eye-point. A scene color for each of the pixels is stored in the color buffer of a frame buffer. Only two pixels ~~P1,P2~~ P1, P2 are shown in detail in the interest of clarity. As shown in FIG. 7A, after step 412, the color for pixel P1 includes P1 scene color. Similarly, the color for pixel P2 includes P2 scene color.

The paragraph spanning page 15, line 21, through page 16, line 4 has been amended as shown below:

Pass One begins (step 520) and proceeds to step 521. In step 521, a texture coordinate generator is initialized. For example, in one ~~an OpenGL~~ implementation, using, e.g., OPENGL® software, a texgen is initialized. The texgen is a functionality that enables a coordinate value to be converted to a scaled texture coordinate value. A one-dimensional texture having texels with values that range from a minimum value to a maximum value is also loaded. For example, the texel values can be 8-bit texels and range from 0 to 255. In other examples, a smaller or greater number of bits are used to decrease or increase resolution. In step 522, a stencil buffer is initialized to 0. For example, a stencil bit associated with each pixel is set to 0.

A12

The paragraph on page 16, spanning lines 14-24, has been amended as shown below:

One example implementation for setting the stencil value is shown in FIG. 6 (steps 610-630). Steps 610-630 are carried out for each pixel in a frame corresponding to a scene to be rendered. In step 610, a depth test is performed on each corresponding front face and back face boundary points in the volume object data. ~~If a~~ When the depth pass passes, control proceeds to step 528. ~~If a~~ When the depth test fails, this indicates a condition where the back face or front face boundary point is located behind a respected pixel. The stencil bit value is then incremented by one at the occurrence of each back face boundary point (step 620). The stencil bit value is decremented at the occurrence of each front face boundary point (step 630). In this way, ~~step~~ steps 620 and 630 act to set a stencil bit value that indicates whether a ~~respected~~ respective pixel is inside a volume region.

A13

The paragraph on page 17, spanning lines 1-7, has been amended as shown below:

For example, as shown in FIG. 7B, pixel P1 is located outside of all of the fog objects 710-730. The depth of pixel P1 relative to the eyepoint is in between fog object 720 and fog object 730. Accordingly, boundary points in the furthest fog object 730 from the eye will fail a depth test with respect to pixel P1. Step 620 will increment (or add to) the stencil value by 1 at the occurrence of the back face boundary point B6. Step 630 will decrement (or subtract from) the stencil bit value by 1 at the occurrence of front face boundary point F6.

A14

The paragraph on page 17, spanning lines 8-14, has been amended as shown below:

Pixel P2 is located at a depth inside fog object 720. Accordingly, the stencil bit value is incremented at the occurrences of back face boundary points B4 and B5 in step 620. The stencil bit value is decremented at the occurrence of front face boundary point F5 in step 630. As a result, the final stencil bit value after steps 620 and 630 is equal to one for pixel P2, indicating that the pixel is inside fog object 720. The final result of ~~step~~ steps 620 and 630 for pixel P1 equals zero, indicating that pixel P1 is outside of the fog objects 710-730.

A15

The paragraph spanning page 19, line 20 to page 20, line 2, has been amended as shown below:

where the contents of the alpha buffer for pixel P2 is equal to the sum of the magnitude between boundary point B1 and Z and the magnitude of the distance between ~~boundary~~ point P2 for pixel P2 and Z, scaled by a scale factor. The scale factor is equal to a fogScale value divided by the magnitude of the distance between the minimum distance plane Z and maximum distance plane M. On the other hand, after step 534, the content of the alpha buffer for pixel P1 is not changed since the stencil bit is set equal to 0.

A16

The paragraph on page 21, spanning lines 4-8, has been amended as shown below:

which equals a sum of the magnitudes of the distances between the boundary points F2 and B2 and the boundary points F4 and B3, scaled by a scale factor. ~~The scale factor~~ The scale factor is equal to a fogScale value divided by the magnitude of the distance between the minimum distance plane Z and maximum distance plane M.